Nonvolatile Memory Analysis

Diagnostic aid page application

Fri, Oct 14, 1994

Local station system software supports a memory-resident read-only “file” system that allows for downloading “named programs.” These programs can be page applications, or local applications or data files. The only write provision is that of writing a new version of an entire file. This can happen by using the Download Page for copying a program between stations or downloading S-records via the serial port. It can also happen via the TFTP server, commonly used to initially download a program from the Macintosh-based MPW development environment. Nonvolatile memory is used for storing these “files” of program/data, in order that they can survive a power-off condition. Because this area is a very long-term storage, corruption can occur from not-quite-bug-free programs. Part of this potential problem is alleviated by use of a checksum of the “file” contents. The checksum is checked when the file is read out. But there have been occasions when allocated space has been in a state other than “free”. The diagnostic tool herein described is designed to help analyze the integrity of this nonvolatile file system.

Each file is referenced by an entry in the file directory, which consists of the entries in the CODES table. Each 32-byte entry specifies an 8-character name, such as LOOPECHO or PAGEPARM, a size, a checksum, a download ptr, an execution ptr, and a version date. The nonvolatile memory area itself includes a header that specifies a ptr to the linked list of free spaces, the total size of the area, and the maximum contiguous free space. At system reset time, all free spaces are coalesced into a single free space. A free space includes in its first 8 bytes two longwords, one a ptr to the next free space (or NIL) and the other the size of its own free space. An allocated block includes a header that specifies the allocated size of that block.

The plan for this diagnostic tool is to scan through the contents of the nonvolatile area, identifying the type of entry, and producing a list of ranges of addresses containing each type. One can select to look for allocated blocks, or free spaces, or both. The resultant list will show holes where a non-matching type is found.

A free space is identified by its inclusion in the free space linked list. An allocated space is identified by an entry in the CODES table whose download ptr references it. An entry that is neither free nor allocated is unavailable for use. Such areas need to be identified so that they can be converted into free spaces for subsequent use. This diagnostic tool is especially aimed at finding such unavailable areas. A nonvolatile area that is unknown may be converted into a free space. For the moment, this must be done manually.

Display layout

```
(put picture here)
```

Enter the node# of the station of interest. An interrupt causes the following sequence of actions:

1. Request global variable at $F50, the ptr to the nonvolatile area. Request non-volatile area header. Request size of CODES table from its table directory entry (#9). Request first part of CODES table. (Minimum size = 64 entries.) Show nonvolatile area ptr on display

2. Request rest of CODES table, using info about its size, for local reference. When collected, show number of valid entries after CODES=.

3. Request first two longwords of each free space block in nonvolatile area, and save for local reference, following linked list of free blocks. Display total #free blocks after FREE=.

4. Request first two longwords of each block in area, beginning with base address + 0010. For each such block, determine type of block, whether free (in free space list), allocated (has ptr in CODES table), or unknown. If matches requested type of interest (A, F or – on second row), then add into range coalescing logic and update display. Update #allocated blocks found after ALLOC=. Update #unknown spaces after UNKN=.

Diagnostic results

Many stations were found to have unknown blocks. Almost all of them were blocks with no data inside. This could happen by use of the download page to start a transfer from S-records via a serial port, when the transfer was not completed. Such a transfer could be targeted at a group address such as, say, all Linac stations. This leaves an invalid CODES table entry that, when replaced by a subsequent successful transfer of the same program, leaves an allocated block that is orphaned; *i.e.*, no CODES table ptr references it.

In order to "fix" this problem, find a free entry in the CODES table, and fill it with an entry using the name AAAAAAAA, say. Enter the size – 8 as found from the bottom line of this diagnostic page, and also enter the download ptr as the address + 8 from the bottom line. Go to the download page, display entries starting with A, and you will get this single entry. Type zeros on top of the size given and interrupt immediately following the zeroed size value. The problem block will be freed. Return to the diagnostic page to again check for more problems. This method assures that consecutive free blocks will be coalesced and the linked list maintained in order.